Energy, Mines and Resources Canada | Énergie, Mines et Ressources Canada

CANMET

Canada Centre for Mineral and Energy Technology | Centre canadien de la technologie des minéraux et de l'énergie

ON THE APPLICATION OF KNOWLEDGE ENGINEERING TO THE TRUCK DISPATCH/SIMULATION PROBLEM

N.J. Stuart, R.J. Kolada

April, 1987

This document is an unrevised interim report prepared primarily for discussion and publication reporting the progress of a research project. Publication does not necessarily imply that the opinions or conclusions are those of the Canada Centre for Mineral and Energy Technology (CANMET)

Ce document est un rapport provisoire non-révisé et rédigé principalement pour fin de discussion et publication documentation des travaux de recherche. La publication n'implique nullement que les opinions ou conclusions sont celles du Centre canadien de la technologie des minéraux et de l'énergie (CANMET)

MICROMEDIA

17 pp

COAL RESEARCH LABORATORIES
DIVISION REPORT CRL 87-51 (TR)

87-05167/

ON THE APPLICATION OF KNOWLEDGE ENGINEERING TO THE TRUCK DISPATCH/SIMULATION PROBLEM

by

N.J. Stuart*, R.J. Kolada**

ABSTRACT

Problems associated with developing a new generation of equipment management software are discussed and the option of using the 'Prolog' language discussed. The concept of knowledge engineering is introduced -interviewing persons knowledgeable about a problem, coding their knowledge, writing software to access that knowledge and finally checking with the original 'experts' as to whether they have been understood correctly.

The nature of simple Prolog programming is described with examples. A sample program is described which models truck operations at an imaginary mine based on the type of knowledge that might be obtained from various mine personnel.

Finally other applications of knowledge engineering in mining are reviewed. This report constitutes the notes for an informal seminar to be given in the Coal Research Center, Devon, Alberta.

---

*N.J. Stuart, Research Scientist, Energy Mines and Resources Canada, CANMET, P.O. Bag 1280, Devon, Alberta, Canada, T0C 1E0.
**R.J. Kolada, Research Scientist, Energy Mines and Resources Canada, CANMET, P.O. Bag 1280, Devon, Alberta, Canada, T0C 1E0.

UTILISATION DES BASES DE CONNAISSANCE POUR RÉSOUDRE LES PROBLÈMES DE SIMULATION DE LA RÉPARTITION DES CAMIONS

par

N.J. Stuart*, R.J. Kolada**

RÉSUMÉ

Les problèmes encourus lors de la mise au point d'une nouvelle génération de logiciel pour réaliser la gestion d'équipement sont examinés dans le présent rapport ainsi que la possibilité d'utiliser le langage "Prolog". Les bases de connaissance sont préparées à partir de consultations avec des experts sur le sujet traité, du codage des informations recueillies, de la préparation de logiciels en vue d'avoir accès aux informations et enfin, de la vérification des informations par les experts consultés au début de l'élaboration du logiciel, dans le but de vérifier si les renseignements reçus ont été compilés correctement.

Dans le présent rapport, la nature de la programmation Prolog est décrite à l'aide d'exemples. On y décrit, en outre, un programme-échantillon qui permet de simuler les opérations d'un camion dans une mine imaginaire. Le programme est basé sur le genre de renseignements qui pourraient provenir des différents membres du personnel travaillant dans une mine.

Enfin, d'autres utilisations des bases de connaissance dans le domaine minier sont étudiées. Ce rapport forme l'ensemble des notes destinées à un exposé dans le cadre d'un séminaire informel qui aura lieu au Centre de recherche sur le charbon de Devon en Alberta.

---

*N.J. Stuart, Chercheur scientifique, Énergie, Mines et Ressources Canada, CANMET, C.P. 1280, Devon, Alberta, Canada, T0C 1E0.
**R.J. Kolada, Chercheur scientifique, Énergie, Mines et Ressources Canada, CANMET, C.P. 1280, Devon, Alberta, Canada, T0C 1E0.

CONTENTS

	Page
ABSTRACT	i
RESUME	iii
CONTENTS	v
INTRODUCTION	1
SML AND THE TRUCK PROBLEM	1
History of the Project	1
Current Plans and Doubts	1
OPTIONS IN SOFTWARE DEVELOPMENT	3
PROLOG, AI AND KNOWLEDGE ENGINEERING	4
KNOWLEDGE ENGINEERING	8
THE PROGRAM	8
The Simulation	11
OTHER APPLICATIONS IN MINING	11
CONCLUSIONS	12
REFERENCES	13

INTRODUCTION

This report is a preliminary review of the possibility of using rule based programming to dispatch and simulate trucks. The idea is that when you ask an engineer how trucks should be dispatched it is to be expected that the condition will be described more in terms of logical rules than in terms of numerical optimization. The approach of coding such rules is referred to as knowledge engineering.

SML AND THE TRUCK PROBLEM

History of the Project

For many years SML (formerly the Calgary Coal Research Lab) has been involved in operations research with the aim of producing computer models of operations that would be found useable by the industry. The programming for this project has for the most part been done by the National Research Council in Ottawa who have cooperated in this research. The early work included field time studies (using note book and stopwatch), systems analysis, evaluation of existing programs made available by industry and some work on new programs. With the creation of the 'Surface Mining Lab' CANMET'S commitment to this project increased by several man years and with it the urgency with which useable results were required. Further, the increase in the number of available commercial programs meant that if CANMET cannot offer something extra or different it should not be in software development at all.

Current Plans and Doubts

During the last year CANMET'S relationship with NRC has been reviewed. The position of NRC appears to have become one where it cannot be expected to devote large amounts of man hours to a project and receive only a secondary credit from CANMET in return.

The size of the project appeared to mushroom, not because the technical aspects of simulation increased much but because three factors appeared to be essential:-

1) The user interface. Making the program understandable by (e.g.) mine foremen and also flexible enough to handle complex problems. Friendly user interfaces and quality

documentation had been sadly lacking in the software produced to date (perhaps because they were still in the development stage). There is, no doubt, however that to gain any degree of street credibility a program has to be seen as a producer of results rather than a consumer of one's time. Failure to treat this as a serious issue would jeopardize the project.

2) Multiple and complex uses. When stochastic modelling of truck-shovel operations were first considered by the lab the concept was inflexible without reprogramming -you set up the variables you run the model, you look at the results. Models now under consideration are to answer practical questions that might in many cases require multiple runs of the model to show the effects of e.g. varying rolling resistance on productivity or to evaluate the likely productivity given known physical availabilities for each machine, or to determine the best fixed assignment of a set of trucks found to be available on a given day. It may be that such multiple runs are rather slow and some means of storing results and zeroing in on optimum solutions is desirable.

3) Various Inputs - Various Outputs. Such a program might be used to design a fleet from scratch (increasingly unlikely as the prospect of new mines diminishes) or to advise existing mines on the effects of changes. In the former case one would have to rely entirely on calculated truck times. In the latter one might like to use, at least in part, truck transit times derived from time study at the mine. There are many programs that might be used to predict truck times and a variety of devices used to obtain time study data. It is extremely desirable that the program is capable of supporting a variety of input data modes and also of producing output graphics on a variety of graphics terminals printers and plotters.

4) Interface to other applications. Serious consideration must be given to building software interfaces to products already used by mines:- geological modelling software, truck monitoring systems such as "OBDAS", or financial modelling

packages. This raises serious questions about how such a choice can be made.

5) Dispatching. Any contemplated modelling system must today include provision for modelling the effect of automated truck dispatching systems.

6) Modularity. It is also clear that everything cannot be done at once.Parts must be created in the order in which they are most needed and the way must be kept open to add additional modules for equipment not currently contemplated. Eventually it is envisaged that the system would include the ability to model combinations of trucks, conveyors, draglines, bucketwheels, easiminers etc. in any combination.

7) Development punctuality. Managing such a project is extremely complex. The overall schema must be developed, then individual modules created and extensively tested and delivered on time. Further some means of distribution must be found to incorporate any training required.

The above can be referred to as a complexity explosion. There is an enormous gulf between a research program (that in the hands of its developer, can simulate whatever is required and be adapted to talk to any hardware or software product) and a mature package that can be of genuine application to mine personnel without assistance. The difference may be measured in hundreds of thousands of dollars. With this in mind this seminar is an exploration of a different philosophy to software design and solving the problems of equipment management.

OPTIONS IN SOFTWARE DEVELOPMENT

Software development to date has been in FORTRAN 77. In North America this remains the language of choice amongst engineers. More modern languages such as Modula 2, C or Pascal provide additional features that would make model development easier in some cases and also promote more structured and maintainable code. Most current computer science graduates would choose such a language in preference to FORTRAN; however, the large numbers of engineers who use FORTRAN give it a momentum.

A third option is the use of a specialized simulation language such as SLAM or Simula. One of our colleagues favors the use of the 'SLAM'

language that allows him to put together simulations very quickly indeed. This is a language that is FORTRAN based so that FORTRAN subroutines may be called. Apart from the cost of the package there is no advantage to not using such a system. Indeed the NRC used a set of subroutines called ANEVENT which gave them many of the features of SLAM. These packages save the programmer from having to reinvent software to do the basic functions of discrete event modelling and are particularly strong on setting up networks, producing various random numbers from a variety of distributions and interfacing to continuous type models. If the central problem facing this group was the modelling process then using a specialized simulation language would be the approach taken.

At present it seems possible that CANMET will enter into an agreement with Queen's University to adapt an existing model to meet extended specifications. This has the advantage of low cost, cheap additional development cost and the fact that we can see the core of the program before we enter into the agreement. There are disadvantages in this approach. One ends up with a program that is jointly owned which has been a problem in the past. The technique of starting with a small program and continually extending it may produce convoluted code -errors fixed with patches, documentation produced at different times written in different styles with different nomenclature conventions. The use of non-structured languages and poor programming style would aggravate this problem.

This would seem an expedient option at the moment when so many questions must be raised about the future computing environment at mines in western Canada and there are so many options for software development. However, the authors feel that it is a good time to be considering radically different approaches to the problem. The rest of this report is the exploration of one such approach.

PROLOG, AI AND KNOWLEDGE ENGINEERING

Prolog is a language whose name derives from 'Programming in Logic'. It originates in France though it is generally identified with workers at the Artificial Intelligence unit at Edinburgh. A prolog program can be thought of as a set of facts, rules and definitions. A conventional program starts at the beginning and runs to the end, perhaps changing the order in which things are done according to the data received. Prolog by

contrast is much less 'procedural' -it has to be given a 'goal' and it goes away and examines its database to attempt to find all the solutions to that goal. For example, the following program represents certain geological 'facts':-

```
rockgroup(coal,sedimentary).
rockgroup(oilsand,sedimentary).
rockgroup(ironore,sedimentary).
rockgroup(ironore,metamorphic).
rockgroup(ironore,igneous).
rockgroup(marble,metamorphic).
```

The following 'goals' produce the following results (note that variables start with capital letters so solutions must be found for then).

1) to list all the metamorphic minerals in the database:-

```
GOAL> rockgroup(Possible_mineral_found,metamorphic)

   OUTPUT>
   Possible_mineral_found = ironore
   Possible_mineral_found = marble
   2 solutions.
```

2) To find out possible source rocks of iron ore:-

```
   GOAL>rockgroup(ironore, Iron_source_rocks)

OUTPUT

   Iron_source_rocks = sedimentary
   Iron_source_rocks = metamorphic
   Iron_source_rocks = igneous
   3 solutions
```

The following program includes 'rules'. The symbol :- can be read as 'if'. Text included between /* and */ are comments and are ignored by the compiler. /*information on coal bearing stratigraphy */

```
followed_by(devonian,carboniferous).
followed_by(carboniferous,permian).
followed_by(permian,triassic).
followed_by(triassic,jurassic).
followed_by(jurassic,cretaceous).
followed_by(cretaceous,tertiary).
followed_by(tertiary,quaternary).

/* some rules : */

/* 'Young' is younger than 'Old' if 'Old' is 'followed by' 'young'*/
younger_than(Young,Old):-
        followed_by(Old,Young).

/* this is a recursive definition:-
        'Young' is younger than 'Old' if there exists 'X' such that
'Young' is
        younger than 'X' and 'X' is younger than 'Old'. */
younger_than(Young,Old):-

        followed_by(Old,X),
        younger_than(Young,X).
```

What happens if we ask:

```
GOAL> younger_than(cretaceous,permian)
```

since permian is not followed by cretaceous we move on to the second definition of younger_than. Here it gives a value to X to fulfil:

```
followed_by(permian,X)
```

This can be satisfied by the first definition of younger_than which will try the value of triassic for X. Next having 'instantiated' X as permian we move on to the second part of the second definition of younger_than and consider:

```
younger_than(cretaceous,triassic)
```

We are more or less where we started except that the 2 periods that we are considering are one step nearer to being adjacent in the time scale. younger_than(cretaceous,triassic) does not succeed in the first definition of younger_than so instead it enters the second definition. This time the value of jurassic is given to X and the second part of the second definition becomes

younger_than(cretaceous,jurassic)

this succeeds by the first definition of younger than. Thus younger_than(cretaceous,permian) is found to be true. This definition is called recursive since the second definition of younger_than refers to itself. We can use this program to evaluate more complex goals; for example:-

to determine periods younger than the triassic:-

```
GOAL> younger_than(Young_rocks,triassic)
Young_rocks=jurassic
Young_rocks=cretaceous
Young_rocks=tertiary
3 solutions
```

to determine periods older than the triassic:-

```
GOAL> younger_than(triassic,Old_rocks)
Old_rocks=permian
Old_rocks=carboniferous
Old_rocks=devonian
3 solutions
```

There is an additional point to note here and that is that in a sense we are building a method to infer whether devonian rocks are younger or older than cretaceous ones. One could call it (stretching a point rather) an 'inference engine'. Traditionally the term 'inference engine' refers to programs that verifies or refutes logical statements (e.g. 'all Irishmen without cars eat mutton') from a database of other logical statements (all mutton eaters like beer etc.) . The term here refers to engines that are less general but efficiently solve problems with specific 'topologies'. The above 'topology' is linear -the carboniferous is only followed by one period etc.. A more complex example is the problem of determining the shortest route between two points given a network of roads. This is a problem of

interest to us in the truck program. but has not been addressed in this report.

KNOWLEDGE ENGINEERING

This term is given to the process of turning an expert's knowledge into software. The term 'expert' is very loose here -an expert may be someone who can give you a list of trucks in a mines fleet, someone who can describe hov a mine wishes to mine it's coal or someone who can describe the precise details and advantages of a variety of dispatching algorythms.

This interview or interviews must be turned into a set of coherent rules that can be coded. This includes adding rules that one hopes were not mentioned because they were to obvious, introducing a consistent nomenclature and planning the execution of the program (such as determining which 'rules' will change at run time). These rules are then coded and a working program produced for inspection/use by the mine. Changing/maintaining the code follows the same sequence. This differs from conventional programming in the closer relationship between the comments made by the 'experts' and the actual code -a fact as told by an expert will generally translate into one or more prolog statement.

THE PROGRAM

The philosophy behind this program is based on various commentaries on the dispatching problem (1 and 2). The observation is that a dispatching system is unlikely to make dramatic savings over fixed dispatching for many configurations provided that the fixed dispatching is optimal (which it often isn't). The real benefit would probably be that the trucks being under close supervision; it would eliminate both sloppy management and the truck drivers ability to ignore instructions undetected. However, it is also suggested in the literature that the selection of a truck dispatch algorythm is tricky from a number of standpoints. Firstly, it is subjective. Secondly there may be unexpected side effects -a program designed to maximize truck utilization will, for example, (over) service shovels that have short cycle times. Thirdly 'logical patches' may be required to deal with site specific issues. Changing the logic of such programs is inviting the introduction of gross error. Alternatively the mine operator may be bombarded with optional 'fiddle factors' to input. The

overall impressions of such programs is a misleading impression of objectivity and scientific method.

The program attempted here is supposed to represent a 'knoving program', one that would serve as an advisor to a radio dispatcher (at least initially). One which represents a description from a variety of mine personnel of what they are trying to get the mine to do. Thus the program contains lines that represent such comments as:

" If we get a sudden order on the spot market the mine plan goes out the windov -we do whatever we can to keep the prep plant at full capacity "

" If production is ahead of schedule we would like to make sure that we are on plan in terms of the ratio of extractions from the various producing areas of the mines -and we must check that stripping is keeping pace with production and vice versa".

These are, if you will, the voice of the mine manager or monthly planning engineer speaking to the daily planning engineer whose input we will consider next. The above are the highest level rules and in a sense control lover level rules that are concerned with the details of implementing them. In a sense then these upper level rules select the operating 'mode' of the mine. In part they operate on directly entered information telling the computer that we are nov rushing to complete an order etc.. In part the program will keep track of certain parameters that are used to make decisions. For example, in the above the maximize production mode only applies when the raw coal stock pile is not full. The program can keep an estimate of the stockpile contents based on the trucks that it dispatches to it and the prep plants current capacity. There must also be a means of manually correcting these values periodically as there is no other means of feedback.

The next level down in rules is perhaps the voice of the daily planner. These rules explain in greater detail what the first rules mean in terms of truck dispatching and other rules of the mines operation:-

"We don't send the big trucks to pit c because they have difficulty with the grades"

"We have three levels of panic production: 'total panic' occurs when we are in a rush to fill an order and either the raw coal silo is in danger of becoming empty or we believe that otherwise we would not have

enough trucks to satisfy prep plant demand. Then we send trucks to wherever will be the quickest for them to return with coal of the grade needed even if this means that they queue at the nearer pits. The 'normal panic' situation occurs when there is a rush on for coal but the rav coal silo is not near empty. Under this situation we want to eliminate the situation in the 'total panic' where a truck queues at a near shovel in preference to an idle more distant shovel. The third level of panic occurs when there is a rush of for coal except that the silo can't handle any more -this is really just the same as normal operating conditions" Also included at this level is the representation of the medium term mine plan. The next type of rules are the permanent facts about the mine site such as lists of trucks, rules that give the travel times between places, equipment specifications etc.. In a more complicated version of this program there would be a representation of the road network.

There is a class of rule or fact that behave as variables but that have initial values in the program representing the situation at the start of the shift. This includes production statistics, the position of trucks etc.. Prolog allows us to assert or retract rules during the program. Finally there are classes of rules that make the program 'work'. These may be considered the voice of the programmer. Many of these rules fall are trivial additions of common sense rules that the computer needs to know. For example, we have already entered rules representing time study knowledge. We must now enter rules that state the obvious:

"The time at which a truck arrives at a shovel if dispatched now will be the current time plus the time taken to travel there".

"The time at which the truck starts loading is either the time at which it arrives or the time at which the shovel becomes free -whichever is later". In addition to stating the obvious there are rules that do house keeping jobs such as look at the system clock and convert the time to hours and fractions of an hour; rules that update the rule database (these are different from the rules themselves). In a more sophisticated version of the program this might include a natural language interface.

Finally there is the central rules that make it run. This is a reasonable way to decribe the rules by which a mine should operate but it may be thought slightly strange to leave the fundamental part of the program (that which in Pascal would be the 'main procedure') till last. This is

perhaps the central point about this approach to software design, for we can write any number of 'program' rules to use the other rules in any way we want. The two most important uses are to run a dispatchers adviser program (operating in real time) and to permit simulation. The dispatchers adviser:-

When a truck has dumped its load it radios the dispatcher for instruction as to where it should go. The adviser program will give the dispatcher information on all aspects of the state of the mine and which shovel the program selects as being the best destination. The dispatcher, obviously, has the option of overruling the system and selecting a different truck. This program could be designed to run in real time which is unsuitable for demonstration so a simulated version of this will be presented.

The Simulation

Provided that the rules are complete (i.e. there is no situation for which the program doesn't knov where to dispatch trucks), simulations can easily be constructed. This has merit both for the normal applications of a simulator and as a means of checking the program.

Notes about program development:-

For the most part his program can be viewed as a naive use of prolog. That is there is no attempt to improve software performance by changing the structure. There are two reasons for this; firstly because it was a first attempt at a serious program in prolog and secondly because the idea is to write clear code and make the computer do the work.

OTHER APPLICATIONS IN MINING

There are a number of other areas where it is felt that prolog has potential application to mining:-

Dragline selection/pit design. A project is underway with Manalta Coal to investgate the use of a prolog program to select draglines and their operating configuration.

Industrial controls:- Industrial control software can be written in prolog. Prolog would be suitable for logically complex situations, where problems can be defined best in terms of rules, where the operating conditions of components switches between large numbers of states or modes.

It is believed that under these situations prolog would offer advantages in the production of correct maintainable code. Though prolog can 'do arithmetic' it would be best to write arithmetically intense algorythms in other languages and link the two. The authors have experimented with a data aquisition board using prolog.

Expert systems:- True expert systems may have application in mining. In the nearer term the existence of intelligent databases of equipment data etc. and better written applications software will lead the way.

Data dictionary: Suppose a collection of programs was to be written to gather time study data, predict haulage times, simulate truck operations to predict efficiencies and implement dispatching routines based on this information. Besides the programs there would come to exist a large number of data files with different data structures, dates of creation etc.. If one wanted to know about the time it would take a new type of truck to travel a given route one might want to compare predictions with time study data for existing trucks. I.E. one needs some way of keeping track of what files exist, what they have in them and in what structure. This function is called a 'data dictionary' and could well be implemented in prolog. This is an alternative to using a commercial data base system's data dictionary.

CONCLUSIONS

The approach of knowledge engineering appears to be capable of solving a number of problems for mining situations. It would appear to be strong in the case of logically rather than numerically complex cases. It will be pursued both for the truck dispatch case given here and the problem of choosing dragline configurations. It clearly is good at different things than conventional programming so one does not want to overstate its case, rather to continue to explore its possibilities.

REFERENCES

1) Lizotte, Scoble and Bonates, 'Application of simulation to assess Truck/Shovel Dispatching policies' Mine Equipment Selection Symposium, Calgary 1985.

2) Jizheng Tu, Simulation of a haulage Dispatching System in an open pit mine' Ph.D. thesis, University of Utah, 1984.